```
RUN
 1
 5
Ok
```

Experiment using different values for these arguments until you feel confident with this function.

Converting Between Numbers and Strings [STR$, VAL]

The STR$ and VAL functions are used to convert numbers to strings (STR$) or strings to numbers (VAL). Consider the output from the following program:

```
10 A = 123 : B = 56
20 A$ = STR$(A) : B$ = STR$(B)
30 PRINT A$,B$
40 PRINT A + B
50 PRINT A$ + B$
60 PRINT LEN(A$), LEN(B$)
```

```
RUN
 123          56
 179
 123 56
 4            3
Ok
```

Line 30 prints the strings A$ and B$. The output from line 40 is 179, which is the sum of the numbers represented by the variables A and B. Line 50 shows the concatenation of the two string variables, A$ and B$.

In line 60 the output from LEN(A$) is 4, even though A$ is the string representation of the three-digit number 123. The STR$ function always holds a place for the sign of a number along with the number. When the number 123 was converted to a string with the STR$ function, the position for the sign was assigned to A$ along with the number. In this case, the number is positive so it is not printed. A space was inserted in place of the plus sign; therefore, the length is 4. Again, the length of STR$(X) is always one longer than the number of digits in the number.

The argument for STR$ must be numeric but does not have to be an integer. For example:

```
10 A$ = STR$(253.15)
20 PRINT A$, LEN(A$)

RUN
 253.15         7
Ok
```

The VAL function is the opposite of STR$. It converts a string into a number. Enter and run the following:

```
10 A$ = "94598"
20 A = VAL(A$)
30 PRINT A + 2

RUN
 94600
Ok
```

Line 20 converts the string A$ into its numeric value. The output from line 30 prints the sum of that value and 2. If you tried to add A$ and 2, you would get the message "Type mismatch in line 30".

If the first nonblank character of the string being converted by the VAL function is not a plus sign, a minus sign, or a digit, the VAL function returns a 0. For example, VAL("SAM") would be 0 but would not give an error.

This property of the VAL function can be very convenient. If you wanted to write a program that will keep track of employees with employee numbers, you might have a statement like

```
100 INPUT "Enter employee name or number: ",EMP$
```

You could then determine if the user typed a name or number simply by using VAL(EMP$). If we know that no employee will have number 0, then VAL(EMP$) = 0 means EMP$ is a name; otherwise, VAL(EMP$) is the employee number.

Using the ASCII Character Set [ASC, CHR$]

The acronym ASCII is the name of the numeric code used by most computers to internally represent characters. The ASC function is

and printers use codes 128 through 255 for special purposes, such as graphics.

Let's try one more example with the ASC function: namely, that of finding the ASCII code for the letters in a name. This program could, of course, be used to find the ASCII code of any character.

```
10 INPUT "Type in your name: ".NAM$
20 FOR I = 1 TO LEN(NAM$)
30    C$ = MID$(NAM$,I,1)
40    PRINT USING "'!' is ###";C$:ASC(C$)
50 NEXT I

RUN
Type in your name: EGBERT SNEED
'E' is  69
'G' is  71
'B' is  66
'E' is  69
'R' is  82
'T' is  84
' ' is  32
'S' is  83
'N' is  78
'E' is  69
'E' is  69
'D' is  68
Ok
RUN
Type in your name: egbert
'e' is 101
'g' is 103
'b' is  98
'e' is 101
'r' is 114
't' is 116
Ok
```

Notice in line 30 that the MID$ function (not the LEFT$ function) must be used to select the character at position I of NAM$. On the first time through the loop I is 1, so that C$ is assigned the first character of NAM$. On the second time through the loop I is 2, so that C$ is assigned the second character of NAM$. On the last time through the loop I is LEN(NAM$), so that C$ is assigned the last character of NAM$. Also recall from our discussion of formatting with the PRINT USING command (in Chapter 6) that the (!) mark in line 40 is used to print a string length of 1.

Note that the variable **NAM$** is used instead of the more descriptive variable **NAME$** because NAME is an MBASIC keyword. The NAME statement will be discussed in Chapter 14.

RINGING THE BELL
[^G]

One of the more useful control characters is ASCII character 7, which is CTRL-G. Often called ^G or BELL, this character rings the bell on your terminal. You can check to see if your terminal supports this function by giving the command PRINT CHR$(7) and listening for a sound. This control character can be used by an MBASIC program to get the operator's attention to perform a task or to warn that an error has occurred.

Creating a String of the Same Characters
[STRING$, SPACE$]

The function STRING$ is used to create a string composed of repeated identical characters. The format is STRING$(X, Y$), where *X* represents the length of the string and *Y$* is the character to be used. If *Y$* contains more than one character, only the first character is used. Run the following example:

```
10 A$ = STRING$(5,"*")
20 PRINT A$

RUN
*****
Ok
```

This function is useful to pr. ıt headings for reports or tables. Enter and run the following:

```
10 A$ = STRING$(5,"*")
20 B$ = " DECEMBER REPORT "
30 PRINT A$;B$;A$

RUN
***** DECEMBER REPORT *****
Ok
```